# PIC18 e assembly: il primo programma



In questa pagina vedremo come creare il primo progetto e come usare il simulatore per esplorare la struttura ed il contenuto delle memorie del microcontrollore che stiamo utilizzando. Non faremo uso di circuiti fisici perché è cosa complessa capire se un comportamento imprevisto nasce da un errore hardware oppure software.

### La creazione di un nuovo progetto

Ciascun programma deve essere incluso in un progetto. La procedura per creare un progetto è guidata:

- 1. Inizia ovviamente con *Create New Project*. Le scelte di default (*Microchip embedded* e *Standalone Project*) sono quelle in genere corrette
- 2. Occorre quindi scegliere la variante di PIC18 (è facile sbagliare: molte sigle si assomigliano). L'esempio mostrato in questa pagina utilizza un PIC18F45K22, ma nulla cambia scegliendo un altro PIC18
- 3. Se intendiamo utilizzare circuiti fisici (ma non è il caso qui mostrato) viene richiesto il modello di *In-Circuit Debugger*. In questo primo esempio sceglieremo invece il simulatore (*Simulator*)
- 4. L'assemblatore da utilizzare (in seguito MPASM). Se sono presenti più versioni, potrebbe essere conveniente scegliere quella più aggiornata
- 5. Il nome del progetto

Quando si crea un nuovo programma è necessario creare un nuovo progetto oppure fare una copia del vecchio progetto (tasto destro del mouse sul nome del progetto)

L'immagine seguente mostra le finestre principali di MPLAB X:

- In alto a sinistra, l'elenco dei progetti. Se sono presenti più progetti, quello in uso (*main project*, in grassetto) deve essere selezionato con il tasto destro.
- Nella stessa finestra: l'elenco dei file del progetto (nell'esempio uno solo: main.asm). Progetti semplici sono costituiti da un solo file, progetti più complessi comprendono molti file.
- In alto a destra, il contenuto di un file sorgente (nell'esempio main.asm)
- In basso a sinistra, la *dashboard*, che permette di monitorare l'uso delle risorse nonché di modificare le scelte fatte (per esempio: il tipo di processore)
- In basso a destra, i messaggi nelle varie fasi di compilazione, programmazione ed esecuzione. Eventuali errori (qui non presenti) sono evidenziati in rosso
- In alto le varie icone e menu di compilazione e debug

Durante l'uso vengono aperte anche numerose altre finestre...

MPLAB X IDE v3.50 - Test1-simulatore : default ×												
Eile Edit View Navigate Source Refactor Run Debug Team Tools Window Help Q- Search (Ct												
🔁 🔁 🔩 🍤 🦿 (default	• °	• 🏹 • ¥ 🛛 🚾 🖓 PC: 0x0 n ov z dc c : W:0	)x0 : bank 0 🦷 🛒	- How do I? Keyword(s)								
Projects x Files Classes	Start	Page 🗙 🖭 usb_device_cdc.c 🗙 📓 main.asm 🗴	-)									
	Asm 1 2 3 4 5 6 7 8 9 10 11 12 13 14	Source History @	Image: Construction of the second									
Production image: Optimization:     T      Memory	Notific	ifications Output ×										
i) Usage Symbols disabled. Click to enable		Trace/Profiling × Test1-simulatore (Clear	1, Build,) ×									
<ul> <li>□ Data 3936 (0xF60) bytes</li> <li>□ Program 65536 (0x10000) bytes</li> <li>▼ S Debug Tool</li> <li>③ Simulator</li> <li>④ Click for Simulated Peripherals</li> <li>▼ Debug Resources</li> <li>□ Program BP Used: 0 Free: 1000</li> <li>□ Data BP Used: 0 Free: 1000</li> </ul>		CLEAN SUCCESSFUL (total time: 57ms) make -f nbproject/Makefile-default.mk SUBPR make[1]: Entering directory '/home/vv/MPLAB make -f nbproject/Makefile-default.mk dist oduction.hex make[2]: Entering directory '/home/vv/MPLAB "/opt/microchip/mplabx/v3.50/mpasmx/mpasmx" ction/main.lst\" -e\"build/default/producti n/main.0; \"main.asm\" "/opt/microchip/mplabx/v3.50/mpasmx/mplink"	OJECTS= .build-conf XProjects/Testl-simu /default/production/ IXProjects/Testl-simu -q -p18f26k20 -u -l .on/main.err\" -o\"bu -p18f26k20 -v	latore.X' Test1-simulatore.X.pr latore.X' \"build/default/produ ild/default/productio -m"dist/default/produ 7								
				4:65 INS								

### La memoria

Attraverso il simulatore è possibile esaminare lo stato del microcontrollore. Per esempio, attraverso la voce **Window**  $\rightarrow$  **PIC Memory view**, è possibile esaminare il contenuto:

- dei *File Registers*, la memoria a lettura/scrittura (RAM) utilizzata per contenere le variabili. Nell'esempio qui sotto mostrato è piena di zeri, tranne le due celle con indirizzo 0x043 e 0x126, evidenziate in rosso
- della *Program Memory*, la memoria a sola lettura (Flash) che contiene il codice eseguibile. In questo caso è piena di 1, tranne le prime celle

Notifications Output Program Memory F									File Registers ×										
Q	Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	OF	ASCII	
	000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
_	010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
₹-	020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
L	030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	040	00	00	00	A5	00	00	00	00	00	00	00	00	00	00	00	00		
	050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	OFO	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
	120	00	00	00	00	00	00	22	00	00	00	00	00	00	00	00	00	"	
	130	00	0.0	0.0	100	0.0	100	0.01	0.0	0.0	00	00	0.0	0.0	100	00	0.0		

Notifications Output Program Memory × File Registers

<b>P</b>		Line	Address	Opcode	Label	DisAssy
a	⇔	1	0000	EF03		GOTO 0x6
-		2	0002	F000	s	NOP
		3	0004	0000		NOP
		4	0006	EF03	MAIN PROG 0000	GOTO 0x6
		5	0008	F000		NOP
₽		6	000A	FFFF		NOP
		7	000C	FFFF		NOP
		8	000E	FFFF		NOP
		9	0010	FFFF		NOP
	1	10	0012	FFFF		NOP
		11	0014	FFFF		NOP
		12	0016	FFFF		NOP
		13	0018	FFFF		NOP
		14	001A	FFFF		NOP
		15	001C	FFFF		NOP
	1	16	001E	FFFF		NOP
		17	0020	FFFF		NOP
		18	0022	FFFF		NOP
		19	0024	FFFF		NOP
		20	0026	FFFF		NOP

Si noti che queste due memorie hanno una struttura diversa:

- La prima è una memoria RAM volatile, suddivisa a seconda della versione del PIC, in circa 4000 celle da 8 bit ciascuna
- La seconda è una memoria flash non volatile, suddivisa a seconda della versione del PIC, in circa qualche decina di migliaia di celle da 16 bit ciascuna

## Il primo programma

Scriviamo il primo programma, creando un file con estensione .asm. e salvandolo nella cartella virtuale Source Files. Quanto mostrato non fa assolutamente nulla di utile, ma ci servirà per prendere confidenza con il microcontrollore, l'ambiente di sviluppo.

😒 MPLAB X IDE v4.05 - test_ASM : default			
File Edit View Navigate Source Refactor Pro	oduction De	oug Team Tools Window Help	
👚 🎦 🞴 🌗 🍏 🧖 🛛 default	~ °r •	🎯 • 🕨 • 🖳 • 🚬 • 🖏 🚯 • 💆	PC: 0x0 n ov z dc c : W:0x0 : bank 0 🛒 How do I
Projects × Services Files Classes	🖬 📓 ma	n.asm ×	
E- test_ASM	Asm S	ource History 🛛 🔀 • 🚚 • 🔍 🕻	중 문 팀 다   중 중 중   엘 엘   ● □   幽 글
Header Files	1	cblock 0x00	;Accesso alla RAM (8 bits)
🗄 🛅 Linker Files	2	Somma	;Dichiarazione di una variabile nella m
Source Files	3	endc	
main.asm	4	RES VECT CODE 0x0000	· Indirizzo di Reset programma Program
Elibraries	6	GOTO START	; va all'inizio programma
	7		
	8	MAIN_PROG CODE	; direttiva al Linker di inizio program
	9		
	10	START	
	11	2 2 22	
	12	moviw 0x55	
	14	addlw 0x20	
	15	movwf Somma	
	16	sleep	
	17		
	18	END	

Cosa fa questo programma?

- 1. Istruzione movlw 0x55: MOVe Letteral to W register muove letteralmente nel registro W il numero esadecimale 0x55.
- 2. Istruzione addlw 0x10: ADD Letteral to W register somma letteralmente 0x10 al contenuto del registro W, memorizzando il risultato nel registro W
- 3. Istruzione addlw 0x20: ADD Letteral to W register somma letteralmente 0x20 al contenuto del registro W, memorizzando il risultato nel registro in W

- 4. Istruzione movwf Somma: sposta il contenuto dell'Accumulatore W nella Variabile 'Somma'
- 5. Istruzione sleep: SLEEP addormenta il processore, cioè ne sospende l'attività

Alcune osservazioni:

- è necessario che ciascuna riga di codice abbia almeno uno spazio davanti
- le istruzioni possono essere scritte maiuscole o minuscole, ma in seguito saranno sempre in minuscolo per chiarezza
- la prima e l'ultima riga, in un colore diverso, NON contengono istruzioni, ma direttive Nello specifico indicano che il codice dovrà essere memorizzato a partire dall'indirizzo di memoria 0 (Cblock 0x00) e che dopo l'END non c'è più nulla di significativo
- i numeri esadecimali sono indicati con la sintassi tipica del C, ma, in questo caso, non è strettamente necessario, dato che la base 16 è quella predefinita. Quindi, a meno di impostazioni particolari, scrivere 0x10 è la stessa cosa che scrivere 10. Consiglio vivamente la prima codifica (0x10) in quanto la seconda (10) può facilmente portare ad errori di interpretazione da parte di chi legge il codice

#### Assembliamo il programma

Per assemblare il programma appena scritto occorre utilizzare l'icona evidenziata, osservando nella finestra di output l'assenza di errori.

😒 MPLAB X IDE v4.05 - test_ASM : default			
File Edit View Navigate Source Refactor Product	ion Debug Team Tools Window Help		
👚 🚰 🞴 🖣 🍏 🥥 🕼 🕼 🗠	] T • 😿 • • • • • • • • • • • • • • • • •	💵 PC: 0x0 n ov z dc c : W:0x0 : bank 0 🫒 H	ow do IP Keyword(s)
Projects × Services Files Classes	main. Clean and Build Project (test_ASM)	) (Maiusc+F11)	
test_ASM     test_ASM     test_ASM     test_ASM     test_ASM	Asm Source History	[ 〒 문 급   중 용 명 열 ● □   ≝ 글 ;Accesso alla RAM (8 bits)	
E linker Files Source Files anin.asm tibraries E and Libraries E and Libraries E and Libraries	2       Somma         3       endc         4	;Dichiarazione di una variabile ne ; Indirizzo di Reset programma Pro ; va all'inizio programma ; direttiva al Linker di inizio pr	ila memoria RAM File Register
Output			
Simulator × Trace/Profiling × test_ASM (Cr CLEAN SUCCESSFUL (total time: 225ms) make -f nbproject/Makefile-default.mk SUBPR make[1]: Entering directory 'C:/PIC/test_ASS make -f nbproject/Makefile-default.mk dist make[2]: Entering directory 'C:/PIC/test_ASS "C:\Program Files (x86)(Microchip/MFLAEK/V4 Warning[205] C: PIC/TEST_ASM.X\MAIN.ASM 1 :	ean, Build,) × OJECTS= .build-conf M.X' /default/production/test_ASM.X.production.1 M.X' .oS\mpasmx\mpasmx.exe" -q -p18f45k22 -1"bui Found directive in column 1. (cblock)	hex ild/default/production/main.lst" -e"build/defaul;	t/production/main.err" -o"build/de
MPLINK 5.08, LINKER MPLINK 5.08, LINKER Device Database Version 1.39 Copyright (c) 1999-2011 Microchip Technolog Errors : 0 MP2HEX 5.08, COFF to HEX File Converter Copyright (c) 1998-2011 Microchip Technolog Errors : 0 make[2]: Leaving directory 'C'/NC/toct NeW	y Inc.	-m ulst/ueledit/production/test_asn.X.production	.map -z_nrike_bullu=1 -odist/
make[1]: Leaving directory 'C:/PIC/test_ASM			
BUILD SUCCESSFUL (total time: 7s) Loading code from C:/PIC/test_ASM.X/dist/de Loading completed	fault/production/test_ASM.X.production.hex		

Il programma compilato viene automaticamente caricato nella *Program Memory* del simulatore. Nella figura seguente possiamo osservare

- le cinque istruzioni che abbiamo scritto, ma non le due direttive (colonna *DisAssy*, traducibile con Disassemblato)
- l'indirizzo (*Address*) delle celle di memoria in cui sono state memorizzate (solo indirizzi pari, essendo tutte lunghe 16 bit, cioè quatto cifre esadecimali)
- Il codice binario corrispondente (*Opcode*, codice operativo, le sole istruzioni che un processore può elaborare)

Line	Address	Opcode	Label	DisAssy
\$ 1	0000	EF03		GOTO 0x6
2	0002	F000	1	NOP
3	0004	0000		NOP
4	0006	0E55		MOVLW 0x55
5	0008	0F10		ADDLW 0x10
6	000A	0F20		ADDLW 0x20
7	000C	6E00		MOVWF 0x0, ACCESS
8	000E	0003		SLEEP
9	0010	FFFF		NOP
10	0012	FFFF		NOP
11	0014	FFFF		NOP
12	0016	FFFF		NOP
13	0018	FFFF		NOP

#### Eseguiamo il programma

L'avvio del programma è gestito da una serie di icone la cui descrizione è mostrata scorrendoci sopra il mouse:



Nell'ordine:

- Inizia l'esecuzione del programma
- Ferma definitivamente l'esecuzione del programma
- (in grigio) sospende temporaneamente l'esecuzione del programma
- Resetta il programma (ricomincia dall'indirizzo zero)
- Prosegue l'esecuzione del programma dal punto in cui è stato precedentemente sospeso
- Esegue una singola istruzione (esecuzione passo-passo). In realtà sono una serie di icone simili, dal comportamento un po' diverso (ma intuitivo...)

Se si desidera fermare il programma ad una certa istruzione, è possibile inserire un *breakpoint* (linea rossa nell'immagine seguente), cliccando sul corrispondente numero di riga.

Durante l'esecuzione passo-passo possiamo osservare:

- In alto a destra il contenuto del Program Counter (PC) e del registro W. Inoltre, anche se al momento non serve comprenderne il significato, è segnalato il valore dei flags (o Status Bits) e quale è il banco di memoria in uso
- Nella finestra del codice, evidenziato in rosso, un breakpont ed evidenziata in verde e da una freccia l'istruzione che sta per essere eseguita
- Nella *Program Memory*, la freccia verde indica l'istruzione che sta per essere eseguita

<b>X</b>	MPL	AB X ID	E v4.05 -	test_ASM	l : default											
File	Ed	it View	Navigate	e Source	Refactor Produc	tion De	ebug Team Tools V	Vindow Help								
P	9 2		<b>4</b> 9	C	efault 🔻	T	• 🔯 • 🕨 • 🖳 • 🦞	1 · 🕄 🚯 ·		6	🕆 🕅 🕁					
Pro	ojec	ts × Se	ervices	Files	Classes	m 📓 m	🗑 main.asm 🗙									
<ul> <li>test_ASM</li> <li>Header Files</li> <li>Important Files</li> <li>Inker Files</li> <li>Source Files</li> <li>Imain.asm</li> <li>Ibraries</li> <li>Libraries</li> <li>Loadables</li> </ul>						Asm 1 2 3 4 5 6 7 8 9 10 11 13 ¢ 15 16 17 18	Asm Source History Reference of the second s									
Pro	ogra	am Memo	ory													
*		Line	Address	Opcode	Label	DisAss	У									
Q		1	0000	EF03		GOTO 02	16									
	-	2	0002	F000		NOP										
mile.		3	0004	0000		NOP										
-		4	0006	0E55	START	MOVLW (	)x55									
1	_	5	0008	0F10		ADDLW O	0x10									
<b>_</b>	\$	6	000A	0F20		ADDLW 0	x20									
	-	7	000C	6E00		MOVWE (	x0, ACCESS									
		8	000E	0003		SLEEP										
		9	0010	7777		NOP										
		10	0012	FFFF		NOP										
	10 0012 FFFF N															

Inoltre è possibile osservare e modificare il contenuto della memoria RAM (*File Registers*).

8	MPLAB X	IDE \	/4.05	i - te	st_AS	5M : 1	defau	ult															
File	Edit Vie	w N	avig	ate	Sour	ce R	efact	tor P	rod	uctic	n D	ebug	Tea	am 1	Tools	Wi	ndov	v Help					
P	1 🔁 🖴			9 (		defa	ult			~	T	- 12	1 -	•	<b>9</b>	- 🐴		۰ 🖫 🕻		2 🔘 🖄	🕹 순	<b>S</b>	
Pro	Projects × Services Files Classes 🖬												sm >	<									
Projects ×       Services       Files       Classes       Image: Classes         Image: test_ASM       Image: Classes       Image: Clas							Asm 1 2 3 4 5 6 7 8 9 10 11 13 14 15 ¢ 17 18	Sour	ce cblc Somm RES_ MAIN STAF	His ock ( aa endo VEC: GOT( I_PR( addi addi slev END	tory 0x00 c T C 0 0 C 0 C 0 C 0 C 0 C 0 C 0 C 0 C 0 C	ODE STA ODE x55 x10 x20 omma	C RT	<b>x</b> 0000		Accesso al ;Dichiarazi ; Indirizzo ; va all'in ; direttiva	la RAM one di di Res izio pr al Lir	et prog					
File	Register	5	_	_	_	_	_				_			_	-	_							
8	Address	00	01	02	03	04	05	06	07	08	09	AO	0B	0C	OD	0E	OF	ASCII					
-	000	85	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			-			
-	010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			-			
₽	030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			-			
-	040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00						
	050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			-			
	060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		<mark>.</mark>	7			
	070	00	0.0	00	0.0	00	0.0	0.0	0.0	0.0	0.0	00	0.0	00	0.0	00	0.0	former:	point interest	1			